Separate-and-conquer Regression

Frederik Janssen and Johannes Fürnkranz

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Motivation
2. Separate-and-conquer Rule Learning with continuous class attributes
   - Handling of numerical attributes
   - Regression Rule Learning Heuristic
3. Experimental Setup
4. Optimization of the algorithm
   - Optimization of splitpoint and *left-out*-parameters
   - Optimization of the parameter of the heuristic
5. Results
   - Results on the tuning datasets
   - Results on the test datasets
   - Comparison to REGENDER
6. Discussion and Future Work

Motivation

TECHNISCHE UNIVERSITÄT DARMSTADT

- lack of separate-and-conquer based rule learning algorithms for regression
- simple and elegant technique
- generation of simple rules that are interpretable
- framework for optimizing parameters of heuristics is already developed and can be re-used

Mechanisms to deal with continuous target attributes and adaptations necessary to deal with Regression data

TECHNISCHE UNIVERSITÄT DARMSTADT

- adaptation either by discretizing the numeric outcome and afterwards use standard classification algorithms
  - algorithm used to discretize: P-CLASS (Weiss and Indurkhya, 1995))
  - problem: number of classes has to be known in advance
- or by adapting the separate-and-conquer technique to Regression tasks
  1. find a rule that minimizes the error of the covered examples
  2. predict the mean/median over all these examples
  3. then remove all covered examples and start with the next rule if examples are left

Mechanisms to deal with continuous target attributes and adaptations necessary to deal with Regression data

TECHNISCHE UNIVERSITÄT DARMSTADT

- adaptation either by discretizing the numeric outcome and afterwards use standard classification algorithms
  - algorithm used to discretize: P-CLASS (Weiss and Indurkhya, 1995))
  - problem: number of classes has to be known in advance
- or by adapting the separate-and-conquer technique to Regression tasks
  1. find a rule that minimizes the error of the covered examples
  2. predict the mean/median over all these examples
  3. then remove all covered examples and start with the next rule if examples are left
  - using the mean or the median
    - advantage: easy to interpret
    - disadvantage: increases the error in the prediction because many examples are mapped onto the same value
  - using linear models as prediction
    - advantage: more accurate prediction
    - disadvantage: some of the interpretability is lost

Separate-and-conquer Rule Learning with continuous class attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

- to adapt the separate-and-conquer algorithm it has to be defined
  - how to handle numerical attributes
    - done by clustering instances to reduce splitpoints
  - how to define a good regression rule learning heuristic
    - done by combining the error and the coverage of a rule
  - when to stop the learning process (a decision list was used for classification)
    - done by simply stopping the learning process when a certain percentage of examples are covered by the set of rules (called *left-out*-percentage)

Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

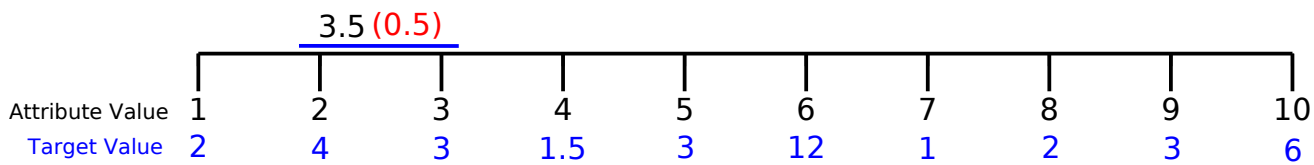
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found

Attribute Value	1	2	3	4	5	6	7	8	9	10
Target Value	2	4	3	1.5	3	12	1	2	3	6

Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

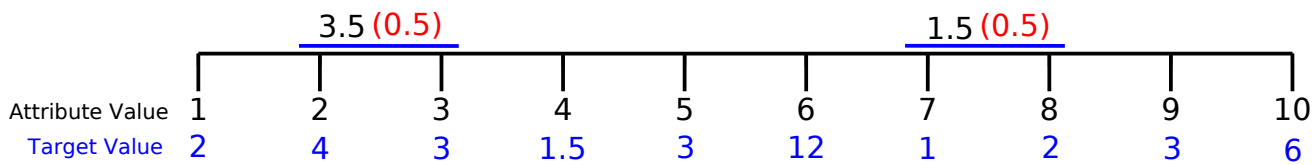
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

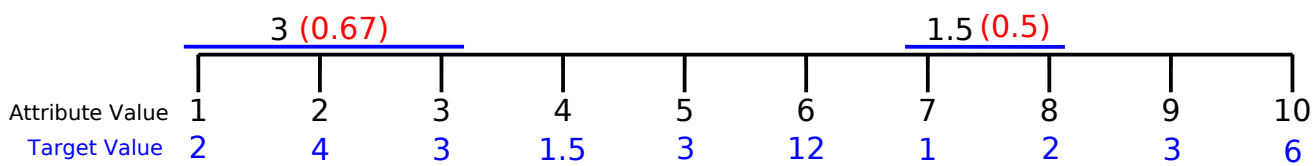
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

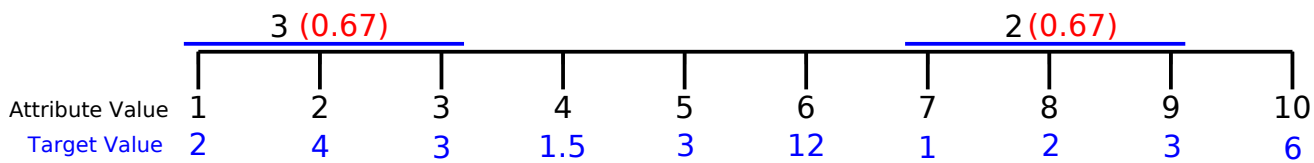
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

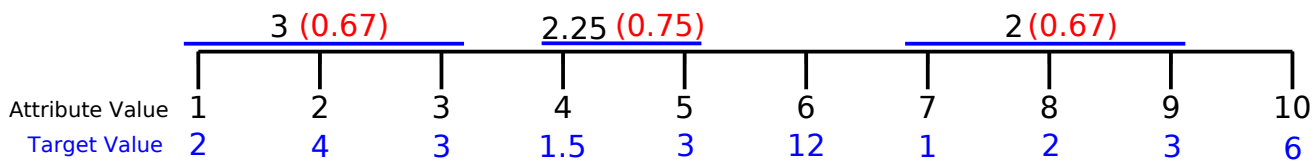
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

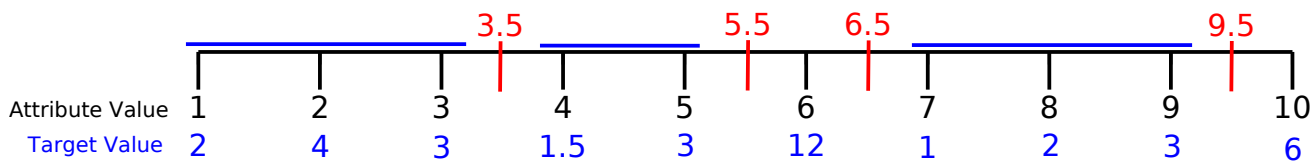
- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Handling of numerical attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

- if all possible splitpoints (those between 2 instances) for all numeric attributes are used the search space explodes
- remedy: do not create all splitpoints but cluster examples together that minimize some error criterion
- and use only the splitpoints between these clusters for each numerical attribute
- Algorithm:
  - sort the examples of the attribute in ascending order
  - remove duplicates by setting the mean over all duplicates as target value
  - merge examples that minimize the mean absolute error
  - stop when the user-given number of clusters is found



Regression measures

TECHNISCHE UNIVERSITÄT DARMSTADT

- Mean Absolute Error $MAE = \frac{1}{m}\sum_{i=1}^{m}|y_i - \bar{y}_i|$
- Root Mean Squared Error $RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \bar{y}_i)^2}$
- Deviation from Mean $def = \frac{1}{m}\sum_{i=1}^{m}(y_i - y')^2$
- Relative Root Mean Squared Error $RRMSE = {}^{RMSE}/_{def}$
- Relative Coverage $relCov = {}^{\text{COVERAGE}(r)}/_{m}$

where m = # of examples in (current) dataset, y_i = true value, $\bar{y}_i$ = predicted value, y' = mean over all instances, r = the current rule

Regression Rule Learning Heuristic

TECHNISCHE UNIVERSITÄT DARMSTADT

- we decided to combine the *RRMSE* and the *relative coverage* of a rule to define the heuristic
- *regression cost measure*: $h_{cm} = \alpha \cdot (1 - RRMSE) + (1 - \alpha) \cdot relCov$
  - α determines a trade-off:
    - setting $\alpha = 0$ results in only accounting for the coverage
    - setting $\alpha = 1$ results in only relying on minimizing the error

Regression Rule Learning Heuristic

TECHNISCHE UNIVERSITÄT DARMSTADT

- we decided to combine the *RRMSE* and the *relative coverage* of a rule to define the heuristic
- *regression cost measure*: $h_{cm} = \alpha \cdot (1 - RRMSE) + (1 - \alpha) \cdot relCov$
  - α determines a trade-off:
    - setting $\alpha = 0$ results in only accounting for the coverage
    - setting $\alpha = 1$ results in only relying on minimizing the error
  - minimizing only the error of the rule would result in a set of rules where each rule covers only a single example ($\rightarrow$ Overfitting)
  - and maximizing only the coverage of a rule would result in a rule that covers all example (nothing is learned then)
  - the best parameter setting lies somewhere in the middle of these two objectives
- note that this heuristic is an adaptation of a classification heuristic (*relative cost measure*), but the heuristics are not directly comparable

Experimental Setup

TECHNISCHE UNIVERSITÄT DARMSTADT

- select 29 datasets from UCI Repository and Luis Torgos Webpage and divide them into 20 sets for tuning and 9 for testing purposes
- compare the tuned algorithm to the *weka*-algorithms M5RULES, SVMREG, LINEARREGRESSION, MLP and to the system REGENDER
- therefore split each of the 20 tuning datasets into 2 folds
- optimize the 3 parameters on each of the folds and test them vice versa
- test them also on the 9 test datasets
- the relative root mean squared error was used for domain-independent evaluation of the algorithms

Optimization procedure

TECHNISCHE UNIVERSITÄT DARMSTADT

- select some parameters for the heuristic and fix the *left-out* to 0 (i.e., all examples are covered)
  - there is evidence that the best value lies somewhere in the middle, thus we included $\{0.4, 0.5, 0.6\}$
  - and the two extremes 0 and 1
  - too much parameters would result in inefficiency
- optimize the splitpoints by testing different values
- fix the splitpoints and optimize the *left-out*-parameter in the same matter
- fix the spiltpoints and the *left-out*-parameter and optimize the heuristic parameter by a greedy search

Optimization of the splitpoint and *left-out*-parameter

TECHNISCHE UNIVERSITÄT DARMSTADT

parameter (splitpoint)	folds 1	folds 2
1	1.0675	1.0540
3	**0.9929**	1.0256
5	1.0132	1.0261
7	1.0067	1.0245
9	0.9992	**1.0209**
11	1.0126	1.0427
19	1.0163	1.0240

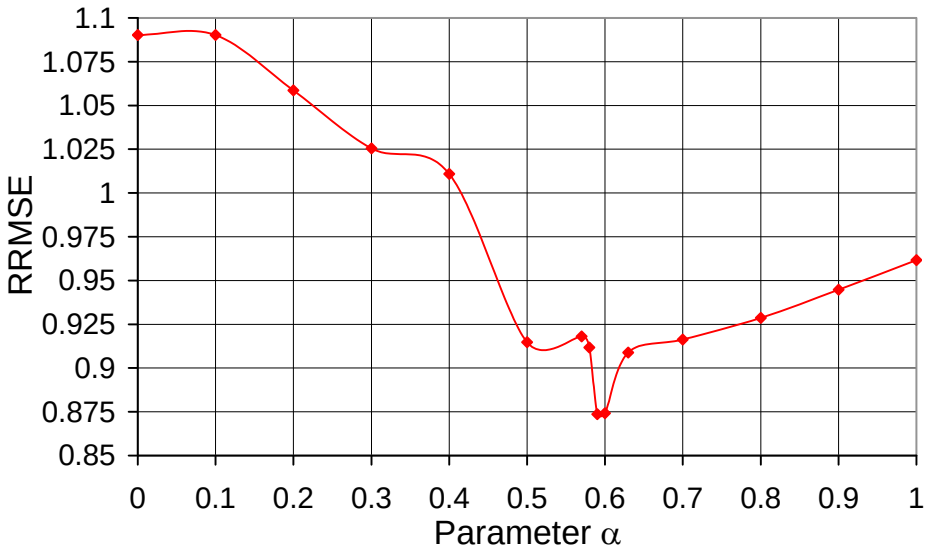
parameter (left-out)	folds 1	folds 2
0	0.9929	1.0209
0.01	0.9787	1.0221
0.02	0.9776	1.0182
0.03	0.9759	1.0156
0.05	0.9739	0.9940
0.1	**0.9704**	0.9835
0.2	0.9736	**0.9701**

- displayed values are averages over the parametrizations $0, 0.4, 0.5, 0.6, 1$ of the heuristic and over the tuning datasets

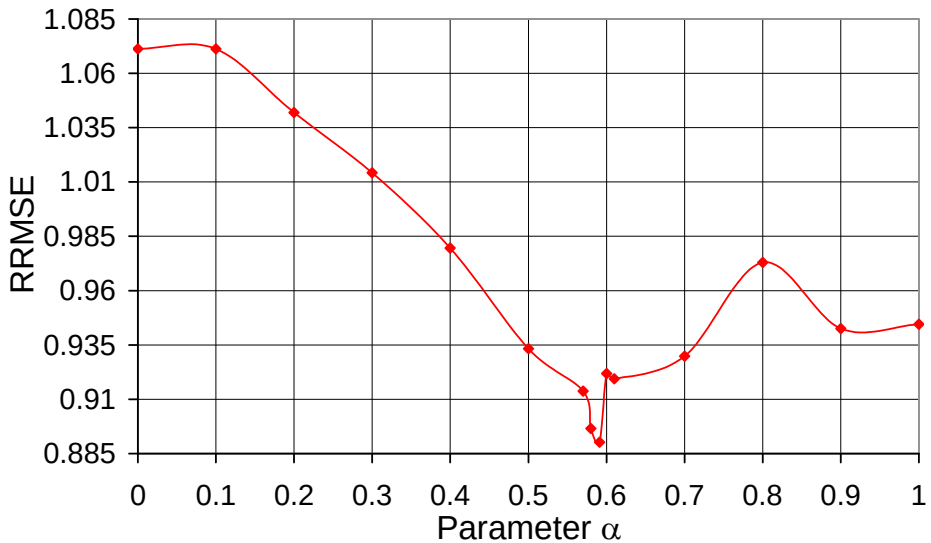
Optimization of the parameter of the heuristic

TECHNISCHE UNIVERSITÄT DARMSTADT

optimized on folds 1



optimized on folds 2

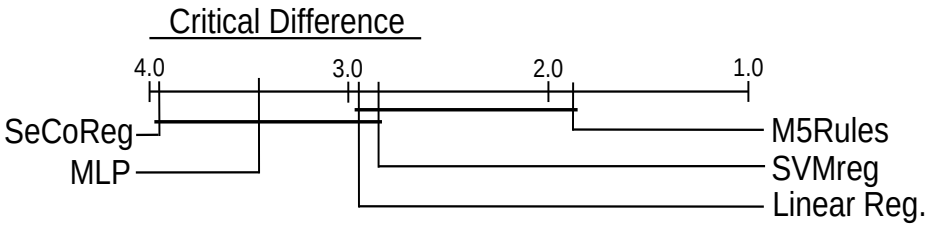


- note that the parameters on the different datasets are nearly the same
- interestingly the best parameter encodes a bias for a lower error (excluding negatives) which is also the case in classification (cf. (Janssen and Fürnkranz, 2010))

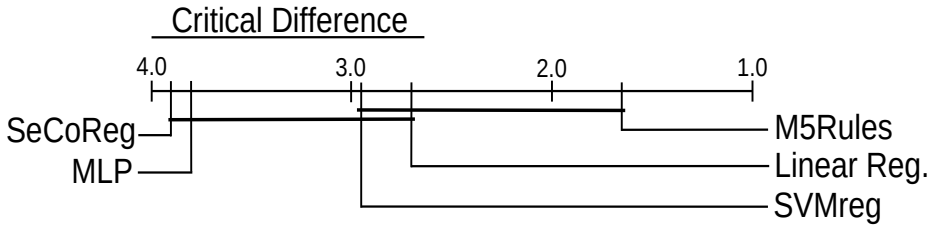
Comparison to *weka*-algorithms on tuning datasets

TECHNISCHE UNIVERSITÄT DARMSTADT

optimized on folds 1



optimized on folds 2



- the tuned algorithm is superior to MLP in average *RRMSE* but inferior to all other algorithms
- only M5RULES is significantly better (at $p = 0.05$) than the tuned algorithm
- note that in the following the tuned algorithms are called SECOREG

Comparison *weka*-algorithms on test datasets

TECHNISCHE UNIVERSITÄT DARMSTADT

dataset	SVMreg	M5Rules	Linear Reg.	MLP	SeCoReg (folds 1)	SeCoReg (folds 2)
auto-horse	**0.32**± 0.08	0.37± 0.14	**0.32**± 0.11	0.34± 0.10	0.52± 0.18	0.61± 0.11
auto93	0.66± 0.12	0.58± 0.19	0.67± 0.20	**0.57**± 0.19	0.65± 0.17	0.85± 0.29
cloud	**0.39**± 0.12	0.42± 0.16	0.40± 0.13	0.62± 0.33	0.61± 0.19	0.67± 0.15
delta-elevators	0.61± 0.01	**0.60**± 0.01	0.61± 0.01	0.63± 0.01	0.78± 0.03	0.77± 0.03
meta	**0.92**± 0.08	1.86± 1.58	2.33± 1.72	1.40± 0.90	1.00± 0.02	1.01± 0.03
r_wpbc	**1.03**± 0.16	1.14± 0.19	1.04± 0.13	2.20± 0.56	1.35± 0.20	1.27± 0.18
stock	0.37± 0.05	**0.14**± 0.03	0.36± 0.04	0.20± 0.04	0.25± 0.03	0.26± 0.04
veteran	**0.93**± 0.15	1.23± 0.61	1.07± 0.36	3.01± 1.78	1.09± 0.22	1.21± 0.33
winequality-red	0.82± 0.03	**0.81**± 0.03	**0.81**± 0.03	0.95± 0.08	0.98± 0.09	0.95± 0.04
averages	0.6739	0.7942	0.8456	1.1017	*0.8040*	*0.8438*

- a Friedman-Test was not rejected at $p = 0.05$, therefore all algorithms do not differ statistically significant
- the tuned algorithms are superior to the LINEARREGRESSION and MLP but inferior to SVMREG and M5RULES
- the algorithm tuned on folds 1 has nearly the same performance as M5RULES

Comparison to RegENDER

TECHNISCHE UNIVERSITÄT DARMSTADT

- RegENDER (Dembczyński *et. al*, 2008) is a rather new rule learning algorithm based on ensembles
- the number of rules in the ensemble has to be defined in advance, therefore the values 10 and 100 were tested and it was run with the same number of rules on each dataset that the tuned algorithms had found

algorithm	avg. rrmse	avg. rank
SeCoReg (folds 1)	**0.82**	**3.0**
SeCoReg (folds 2)	0.85	3.13
RegENDER (10 rules)	0.90	3.88
RegENDER (100 rules)	0.93	3.88
RegENDER (rules from folds 1)	0.92	3.75
RegENDER (rules from folds 2)	0.90	3.38

Discussion and Future Work

TECHNISCHE UNIVERSITÄT DARMSTADT

- our algorithm implements a Separate-and-conquer Regression Rule Learner
- trade-off between consistency and coverage results in same observations as known from classification
- a new splitpoint computing method was introduced
  - only about 10 splitpoints are sufficient for most of the datasets
  - much faster than using all splitpoints
- algorithm is comparable to other systems where some of them are based on rules and some are not
- Future Work
  - the runtime could be improved by many different ways, i.e., by inducing error bounds to perform forward pruning
  - the advantages of predicting linear models could also be exploited
  - other heuristics could be adapted to the Regression task

References

TECHNISCHE UNIVERSITÄT DARMSTADT

- (Weiss and Indurkhya, 1995): S. M. Weiss and N. Indurkhya. Rule-based Machine Learning Methods for Functional Prediction. *Journal of Artificial Intelligence Research*, 3:383-403, 1995.
- Luis Torgos website:
  `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html`
- (Janssen and Fürnkranz, 2010): F. Janssen and J. Fürnkranz. On the Quest for Optimal Rule Learning Heuristics. *Machine Learning*, 78(3): 343-379, 2010.